Iterated Population Based Training with Task-Agnostic Restarts

Alexander Chebykin[1], **Tanja Alderliesten**[2], **Peter A. N. Bosman**[1,3]

[1]Centrum Wiskunde & Informatica, Amsterdam, the Netherlands
[2]Leiden University Medical Center, Leiden, the Netherlands
[3]Delft University of Technology, Delft, the Netherlands
a.chebykin@cwi.nl, t.alderliesten@lumc.nl, peter.bosman@cwi.nl

Abstract

Hyperparameter Optimization (HPO) can lift the burden of tuning hyperparameters (HPs) of neural networks. HPO algorithms from the Population Based Training (PBT) family are efficient thanks to dynamically adjusting HPs every few steps of the weight optimization. Recent results indicate that *the number of steps between HP updates* is an important meta-HP of all PBT variants that can substantially affect their performance. Yet, no method or intuition is available for efficiently setting its value. We introduce Iterated Population Based Training (IPBT), a novel PBT variant that automatically adjusts this HP *via restarts* that reuse weight information in a task-agnostic way and leverage time-varying Bayesian optimization to reinitialize HPs. Evaluation on 8 image classification and reinforcement learning tasks shows that, on average, our algorithm matches or outperforms 5 previous PBT variants and other HPO algorithms (random search, ASHA, SMAC3), without requiring a budget increase or any changes to its HPs. The source code is available at https://github.com/AwesomeLemon/IPBT.

1 Introduction

Machine learning often achieves excellent results thanks to well-tuned hyperparameters (HPs). Hyperparameter Optimization (HPO) algorithms can reduce the time and effort needed for tuning (Feurer and Hutter 2019; Tan et al. 2024).

Efficiency is important for HPO in general but it is vital when the underlying task is expensive, as is the case for neural network training, where often only a few training runs can be afforded. One HPO algorithm that is well-suited for this setting is Population Based Training (PBT) (Jaderberg et al. 2017). PBT draws its efficiency from dynamically adjusting HPs during training. Specifically, a population of networks – each with its own set of HPs – undergoes weight optimization via gradient descent for several steps. After this, poorly performing networks are replaced by copies of better-performing ones, with their corresponding HPs copied and perturbed.

Further efficiency gains can be achieved by replacing random perturbations with Bayesian Optimization (BO), an idea explored by several PBT variants (Parker-Holder et al. 2020, 2021; Wan et al. 2022). Despite the reported improvements, it was recently shown (Chebykin et al. 2025) that none of the PBT variants consistently outperforms others, with substantial variability due to the value of the "step size" HP (i.e., how many weight update steps are performed before HPs are adjusted). The step size varies across papers with no discussion on how it is set for the tasks at hand or how it should be set for unseen tasks. Having to empirically determine a good value for this HP detracts from the utility and efficiency of PBT variants.

We aim to address this issue by having the step size adjusted automatically without sacrificing efficiency. Towards this goal, we introduce Iterated Population Based Training (IPBT), a novel PBT variant that performs several iterations of a PBT-like HPO procedure with a step size that is increased on each restart. A restart is triggered upon hitting diminishing returns, as determined by our novel data-driven mechanism. Crucially for efficiency, information is transferred across restarts in a task-agnostic manner: the partially-trained weights are shrink-perturbed (Ash and Adams 2020) (i.e., reduced in magnitude and injected with noise), while HPs are reinitialized via a time-varying meta BO procedure that learns across restarts. Figure 1 presents an example run of our algorithm.

IPBT builds upon the idea of PBT with restarts as introduced in Bayesian Generational PBT (BG-PBT) (Wan et al. 2022). Key differences of IPBT relative to BG-PBT are the step size adjustment upon restarts, *task-agnostic* weight reuse after a restart, and reinitialization of HPs across restarts via time-varying BO. See Sections 2.2 and 3 for further details.

An important advantage of PBT variants is that they run in parallel, taking approximately the same wall-clock time as training a single network. To uphold this advantage, IPBT is designed not to increase the total number of weight update steps, restarting aggressively to most effectively utilize the limited budget. We explicitly target low-budget HPO, with the budget in our main experiments set to 8 full training runs. Additionally, we assume that no prior information about the task is available, motivating a black-box HPO approach.

We evaluate IPBT on image classification tasks using CIFAR-10/100 (Krizhevsky and Hinton 2009), Fashion-MNIST (Xiao et al. 2017), and TinyImageNet (Stanford CS231N 2017), as well as on reinforcement learning tasks using Brax (Freeman et al. 2021) Humanoid, Hopper, Pusher, Walker. IPBT is compared to 5 previous PBT variants (see Sections 2.2 and 4.1) with untuned and tuned step sizes, as well as Random Search (RS) (Bergstra et al. 2011), Asynchronous Successive Halving Algorithm (ASHA) (Li et al. 2020), and SMAC3 (Lindauer et al. 2022). We additionally

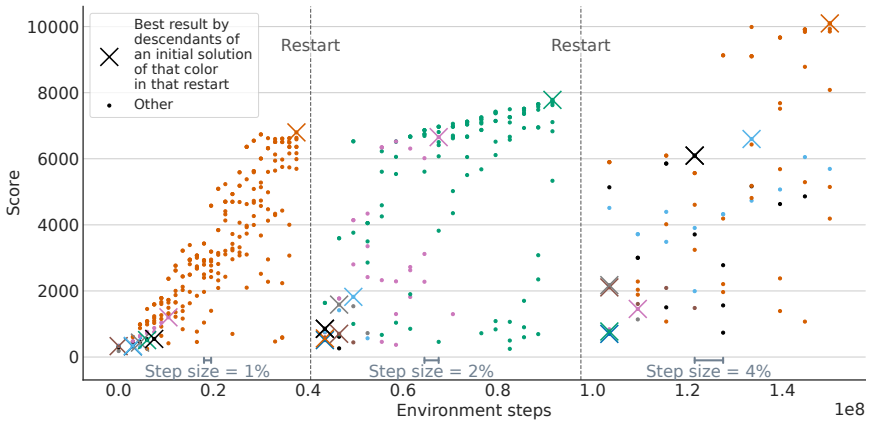


Figure 1: An example run of IPBT on the Humanoid task. Three iterations can be seen. Within each iteration, HPs are dynamically optimized during training via a PBT-like procedure (Section 2.2). If the performance is determined to be stagnating (Section 3.2), a restart is triggered. Upon restart, information from weights and HPs of previous iterations is reused (Section 3.3), and the step size of the PBT-like procedure is increased (Section 3.4). New HPs are predicted via a time-varying meta BO procedure which is trained on the initial HPs of previous iterations as inputs and their descendants' maximum achieved scores as outputs (in the figure, the descendants of each initial solution share its color). The pseudocode of IPBT is provided in Appendix A.

perform a thorough ablation study of the components of IPBT to better understand their influence.

Our main contributions are as follows:

- We introduce IPBT, a novel PBT variant that efficiently adjusts its step size via restarts.
- IPBT is evaluated on 8 image classification and reinforcement learning tasks, where it is compared with 5 previous PBT variants and other HPO algorithms (RS, ASHA, SMAC3) .
- We explore how different methods of reusing information contained in weights and HPs impact performance.

2 Related work

2.1 Efficient algorithms for expensive HPO

BO algorithms are a good option for expensive optimization tasks, including HPO (Snoek et al. 2012; Cowen-Rivers et al. 2022). However, since BO algorithms are typically initialized with as many random samples as the problem dimensionality, they face difficulties in low-budget HPO settings, where the budget in terms of full training runs is often smaller than the dimensionality. Therefore, while efficient algorithms for expensive HPO often have BO components, they pursue additional efficiency in other ways.

PBT variants (BO and non-BO) are efficient thanks to dynamic HPO where HPs are changed during training based on short-term feedback (we discuss these algorithms in Section 2.2). Alternatively, the HPs can be fixed during training but the training itself may be shorter, use a smaller model, or rely on some other proxy. These algorithms are called "multi-fidelity", as the proxy task may be parameterized to be more or less expensive (and accordingly provide information with higher or lower fidelity) (Won et al. 2025).

In Successive Halving (SH) (Jamieson and Talwalkar 2016), N randomly sampled configurations are trained for a number of epochs. After this, all but the best $\frac{1}{\eta}$ configurations are stopped, and the remaining ones continue training for η times as many epochs (default $\eta = 2$). This procedure repeats until the maximum per-configuration budget R is reached. SH avoids wasting time on unpromising configurations but suffers from greediness and does not learn from experience. Hyperband (Li et al. 2018) improves upon SH by running multiple instances of it with different values of (N, R). Hyperband still relies on random sampling, which was replaced with BO in BO Hyperband (BOHB) (Falkner et al. 2018). Awad et al. (2021) showed that replacing BO with differential evolution further improves results, while Lindauer et al. (2022) demonstrated that using random forest as a predictor in BO performs even better, achieving excellent results as a part of the SMAC3 package. Throughout this paper, we use "SMAC3" to refer to the multi-fidelity setup of this package.

In another research line, ASHA (Li et al. 2020) effectively parallelized SH and was shown to outperform SH, Hyperband, and BOHB without any learning, thus making it the most effective approach based purely on random exploration.

2.2 Population Based Training variants

As mentioned previously, PBT (Jaderberg et al. 2017) enables efficient HPO by dynamically adapting HPs while continuously training the weights. The optimization is bilevel, with inner steps updating the weights and outer steps updating the HPs. PBT operates with a population of N solutions where each solution at an outer step t is a pair of HPs and weights, $(\boldsymbol{h}_t^i, \theta_t^i)$. The weights of each solution θ_t^i are trained using the corresponding HPs $\boldsymbol{h}_t^i$ for $\text{step}_{\text{inner}}$ steps (e.g., via gradient descent), after which the weights are evaluated on a validation set. Each of the worst-performing $\lambda\%$ solutions is replaced with a copy of one of the best-performing $\lambda\%$ solutions (thus ensuring propagation of well-performing weights), and the HPs of the copies are explored via random perturbation. A

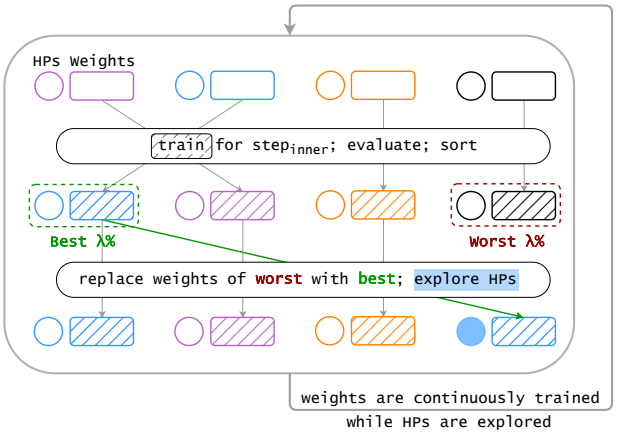


Figure 2: General PBT loop: in each outer step, the weights are trained for $\text{step}_{\text{inner}}$ inner steps and the HPs are explored.

graphical overview of the PBT loop is shown in Figure 2.

In Population Based Bandits (PB2) (Parker-Holder et al. 2020), BO was suggested as a replacement of random perturbation for exploring HPs. A dataset of changes in performance relative to the previous step (y_t^i) is updated after each outer step ($D_t = D_{t-1} \cup \{(y_t^i, t, \boldsymbol{h}_t^i)\}_{i=0}^{N-1}$) and used to fit a Gaussian Process (GP) model. New HPs are then chosen by optimizing an acquisition function that is derived from time-varying GP bandits (Bogunovic et al. 2016) (that are designed for dynamic problems in which the function being optimized changes over time).

PB2-Mix (Parker-Holder et al. 2021) extended the BO approach of PB2 to include categorical HPs. BG-PBT (Wan et al. 2022) further included ordinal HPs while also building upon a BO algorithm (Wan et al. 2021) that is well-suited to mixed-inputs and high-dimensional problems. In BG-PBT, restarts were introduced, with weight information transferred via distillation. We discuss relevant components of BG-PBT when introducing our improvements in Section 3. BG-PBT was strongly focused on enabling Neural Architecture Search (NAS), which we do not address in our work.

Faster Improvement Rate PBT (FIRE-PBT) (Dalibard and Jaderberg 2021) is a non-BO PBT variant that reduces the greediness of PBT by employing several subpopulations running the PBT algorithm with different objectives targeting short- or long-term performance.

The recently-proposed Multiple-Frequencies PBT (MF-PBT) (Doulazmi et al. 2025) algorithm addresses the issue of setting the step size by running several subpopulations with different step sizes. This performs well with large populations but seems infeasible for the ones that are too small to be split into multiple subpopulations of reasonable size. Doulazmi et al. (2025) further observed significant difference in performance between using their default population size 32 and the smaller 16. In contrast, IPBT is designed to efficiently adjust step size with a single small population, relying on restarts and information reuse.

It should be noted that since PBT variants modify HPs during training, a *schedule* of HPs is optimized, which is known to outperform fixed HP values (Tan and Le 2021).

2.3 Optimization with restarts

Restarting is a powerful idea in optimization: when the optimization procedure stagnates, it is restarted from a different initial point. In repeated local search, the new starting point is chosen randomly, while in iterated local search (Lourenço et al. 2003), the new point is chosen so that it is not too far from previously-determined good solutions nor so close as to converge to the same local optimum. A conceptually similar example of such a strategy in deep learning is Stochastic Gradient Descent (SGD) with restarts (Loshchilov and Hutter 2017), where the learning rate is periodically increased (giving SGD an opportunity to escape from a local optimum) while the weights are retained (so the optimization does not deviate too radically from the previous solution).

The weights could also be modified according to the principles of iterated local search. Several algorithms (Sokar et al. 2023; Elsayed et al. 2024) were proposed for modifying previously trained weights to improve downstream optimization. Shrink-perturb (Ash and Adams 2020) is one such algorithm that was shown to perform well in a variety of contexts (Zaidi et al. 2023; Chebykin et al. 2023): the pretrained weights are modified via shrinking (multiplying by an HP λ_{shrink}) and perturbing (adding a random reinitialization of the weights multiplied by an HP γ_{perturb}). This both preserves some information present in the weights and introduces randomness that may enable optimization to find a better solution.

3 Method

3.1 Overview

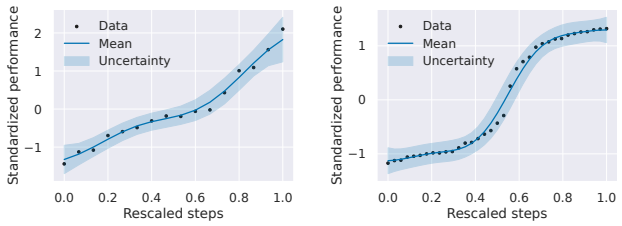
Each iteration of IPBT follows the general PBT procedure in Figure 2. The time-varying BO algorithm of BG-PBT is used to suggest new HPs. We also follow BG-PBT by starting optimization with $N_{\text{multiplier}}$ times more networks than the population size N and keeping only the best-performing N after the first step.

Efficient adjustment of the step size during an IPBT run is achieved by stopping an iteration once the performance stagnates and restarting with a larger step size. This raises the questions of when to start a new iteration (Section 3.2), how to reuse information from the previous one (Section 3.3), and how to adjust the step size (Section 3.4).

3.2 Deciding when to restart

In a black-box setting where nothing is known about the underlying task, one way to select a step size is to empirically try several options. However, fully running a PBT variant several times would reduce efficiency. Efficiency can be preserved by allowing a run with a specific step size to continue only while the performance is strongly improving.

While in BG-PBT the step size was not modified on restarts, the design of its two restart-triggering stopping criteria is relevant. The first one checks that the performance has not improved for t_{patience} outer steps in a row. This, unfortunately, does not take into account the margin of improvement: a glacially but consistently improving run will not be stopped.



(a) (b)

Figure 3: Visualizations of the smoothed standardized scores when a trajectory is (a) improving vs. (b) stagnant.

This is addressed by the second criterion of BG-PBT: restarting after 40 million environment steps have elapsed since the previous restart (corresponding to 26.6% of the budget in the experiments of Wan et al. (2022); for some tasks, this value was set to 60 million steps (40% of the budget)). Since meeting one of the two criteria triggers a restart, the second criterion will eventually terminate the slowly improving run. However, it is not a priori clear how to set the size of the budget that must elapse before a restart is triggered. Additionally, restarting after a fixed budget is inflexible, as a run that improves too slowly cannot be stopped earlier while a strongly improving run cannot be continued for longer.

We propose a data-driven approach that is less susceptible to these issues. The essential questions to be answered are "is the run improving?" and "is it improving too slowly?". To address the first question in the presence of noise, we collect the best performances at each previous outer step, z-score normalize these values, and smooth them via predictive parametric GP regression (Jankowiak et al. 2020) that deals well with heteroscedastic noise. The smoothed value at the current step is compared to that of the previous step, triggering a restart if the current value is not better for t_{patience} consecutive steps.

Even if the scores are improving, the rate of improvement may be too slow. Determining this is difficult in a black-box setting with no information about what a good score or a good improvement rate is (neither of which can be known for novel tasks). Therefore, the judgment needs to be made based on the observed performance within the run. Our second stopping criterion leverages the standardized performance data, triggering a restart if the performance has not improved by a standard deviation over the last t_{interval} outer steps. This is task-agnostic and allows the data to speak for itself: if the scores steadily improve, the difference of the current score with that of t_{interval} steps ago is large (allowing the run to continue while the steady gains persist). If, however, the gains are large at some point but later diminish, the performance t_{interval} steps ago will eventually be within a standard deviation, since standardization is done based on the entire trajectory (which incorporates information about the possibility of fast improvements). Figure 3 illustrates both cases.

3.3 Reusing information from previous iterations

Weights Leveraging information contained in partially-trained weights is crucial for efficiency of the PBT paradigm. Introducing restarts requires figuring out how to transfer this information across them. The two simplest approaches are to either copy the weights from the previous iteration (thus treating the restart as any other outer step of a PBT) or to reinitialize them randomly (as the weights could have caused the stagnation of the previous iteration).

BG-PBT instead relies on distillation to transfer useful behavior from the previous iteration into the next one. This approach allows information transfer even across different architectures but comes with a serious downside: the distillation procedure and its HPs needs to be adapted to the task at hand. Clearly, a fixed distillation procedure cannot be applied across deep learning settings as different as reinforcement learning and image classification. For black-box HPO, a more general approach is desired.

Such an approach should work with the weights directly, since they are the only commonality in all deep learning scenarios. It should both preserve useful information present in the weights and enable downstream optimization to escape from their basin of attraction.

As described in Section 2.3, shrink-perturb (Ash and Adams 2020) is a straightforward method that consists of copying the weights (multiplied by a constant) and adding noise to them. It is general, operates on weights directly, and both preserves information and injects noise. We integrate shrink-perturb in our algorithm as a method to modify the weights from the previous iteration after a restart. To improve chances of a successful perturbation, before applying shrink-perturb, weights outside the best $\lambda\%$ of the population are replaced with copies of the weights of the best $\lambda\%$.

Additionally, we must consider that weights may end up in unfavorable regions of the optimization space which cannot be escaped via shrink-perturb. To avoid performance degradation in such settings, we randomly reinitialize the weights of half of the population upon restart (instead of shrink-perturbing them).

Hyperparameters After restarting, the HPs of the solutions in the new population need to be set. Sampling them randomly neglects the previous experience. On the other hand, relying on past results could also be harmful, since the weights are trained continuously and HP values that were good initially may not be good for a later stage of the training.

In BG-PBT, an auxiliary global BO procedure was proposed. Firstly, a surrogate S_i is fit on the pairs of (HPs, score) at the latest iteration i. Secondly, the best HPs from previous iterations are found and evaluated using S_i. The dataset of these HPs and evaluations is used to fit an auxiliary surrogate S_A. Finally, $10 \cdot N$ random configurations are sampled, out of which N with the best values of an acquisition function (computed based on S_A) are kept. This procedure does not take all the historical trajectories into account (only one from each restart) and uses performance predicted by one model to fit another (which is likely to bias the results).

We propose an alternative approach, specifically, to perform time-varying BO *at the level of restarts*. That is, the

HPs at the start of each iteration are optimized to maximize long-term performance within that iteration. This introduces an additional level of dynamic optimization to the PBT setup: while BO optimizes HPs within an iteration, meta BO optimizes the initial HPs across iterations.

Since HP values that lead to good results after a restart are expected to change as the weights are trained, it is reasonable to reuse the time-varying BO procedure of BG-PBT, fitting it on the long-term performance of initial HPs. To estimate the performance of a set of initial HPs, we track the history of the weights initially trained with them (before potentially switching to other HPs). The best performance achieved by any of these weights is taken as the long-term performance of the initial HPs. Many initial weights are removed from the population relatively quickly, resulting in shorter-term (and worse) performance compared to the best weights. We consider this desirable, as it steers HPO away from the initial HPs that lead to uncompetitive weights.

This procedure aims to learn from past information, which, however, may mislead the algorithm and bias it toward poorly performing local optima. The exploration performed by BO is limited, which is valuable if its model of the problem is correct, but in a black-box setting we must be prepared for any problem, including those that violate the assumptions made by BO. Therefore, similarly to how we randomly reinitialize the weights of half the population, we also sample random HPs for half the population (with sampling independent for weights and HPs).

3.4 Adjusting step size

BG-PBT introduced a decreasing step size schedule for certain tasks, which were selected manually because fixed-step-size BG-PBT performed poorly on them. It is not clear how such tasks could be identified in advance. Additionally, a deterministic schedule gives the algorithm no opportunity to retain a good step size for longer.

Whereas BG-PBT starts with a large step size, we propose starting with a small step size. An intuitive choice is the minimum reasonable amount of weight updates (e.g., an epoch), or 1% of the budget. Starting with a small step size leads to better use of budget in terms of inner steps: as HPs are updated more frequently, fast gains can be made. At the same time, restarts with larger step sizes enable the algorithm to eventually target longer-term performance.

In IPBT, the step size is doubled on each restart. This exponential growth allows larger step sizes to be reached even with limited budget. Alternatively, linear growth can be considered, with the step size increased from $\text{step}_{\text{inner}}$ to $2 \cdot \text{step}_{\text{inner}}$ to $3 \cdot \text{step}_{\text{inner}}$, etc.

4 Experiments and Results

4.1 Setup

The performance of IPBT is evaluated on the 8 image classification and reinforcement learning tasks that were used by Chebykin et al. (2025) to benchmark PBT variants. The setup of the tasks is followed exactly and the large search spaces are used. The classification tasks entail maximizing accuracy of a ResNet-12 (He et al. 2016) on CIFAR-10/100 (Krizhevsky and Hinton 2009) and Fashion-MNIST (Xiao et al. 2017) datasets and of a ConvNeXt-T (Liu et al. 2022) on Tiny ImageNet (Stanford CS231N 2017). In the reinforcement learning tasks, the score of a proximal policy optimization (Schulman et al. 2017) agent is maximized on the Humanoid, Hopper, Pusher, and Walker tasks from Brax (Freeman et al. 2021). See Appendix B for further details.

Across tasks, the attainable performance scores span different ranges. To enable easy comparison of general algorithm performance, we largely follow the methodology of Agarwal et al. (2021). Specifically, we repeat each experiment 8 times, normalize performance per task across all algorithms, and use stratified bootstrapping to estimate the Interquartile Mean (IQM) and the 95% Confidence Interval (CI). For statistical significance testing, a paired stratified bootstrap test (Efron and Tibshirani 1993) with Holm correction (Holm 1979) is used with $\alpha = 0.05$, as described in Appendix C.

We compare IPBT to the 5 PBT variants available in PBT-Zoo (Chebykin et al. 2025): PBT, PB2, PB2-Mix, BG-PBT, and FIRE-PBT (see Section 2.2 and the original publications for details).

Since we are interested in efficient low-budget HPO, the population size of IPBT and PBT variants is set to 8. Although IPBT introduces multiple components with additional HPs, these are designed to work well out of the box and therefore remain fixed across all tasks (see Appendix D for more details on the HPs). The default values and the components of IPBT are evaluated in our ablation studies (with performance evaluated on 4 tasks: CIFAR-10/100, Humanoid, and Hopper). To ensure unbiased performance estimates, the main experiments were run with seeds different from those used in the ablation studies.

IPBT is run with an initial step size of 1% of the budget (that is automatically adjusted). Other PBT variants are run with the step sizes from (Chebykin et al. 2025): 1%, 3.3%, 10%, 33.3%. For each PBT baseline and for each task, the results across all steps are pooled to estimate untuned performance, while the results corresponding to the step size yielding the maximum performance are considered the tuned performance.

We additionally run non-PBT HPO algorithms: RS as the standard baseline, ASHA (Li et al. 2020) as an efficient multi-fidelity extension of RS, and SMAC3 (Lindauer et al. 2022) as a BO multi-fidelity algorithm. Unlike PBT variants, these algorithms cannot directly search for hyperparameter schedules, placing them at a disadvantage. To address this, we extend their search spaces to include parameters of a cosine learning rate schedule with restarts (Loshchilov and Hutter 2017), thus allowing them to potentially find learning rate schedules similar to that of IPBT. The baseline algorithms are run with default HPs and the same budget as IPBT. See Appendix E for further implementation details.

4.2 Overall performance

We start by comparing IPBT to previous PBT variants in the one-shot setting, where the step size of each of the PBT variants is not tuned. Figure 4 shows that IPBT strongly outperforms previous PBT variants, clearly demonstrating its utility for dynamic HPO without per-task tuning of the

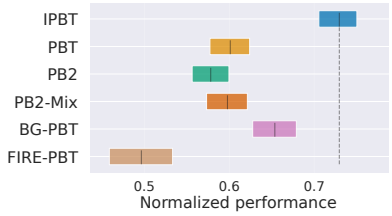


Figure 4: Normalized performance across 8 tasks (IQM and CI) of IPBT and *untuned* PBT variants.

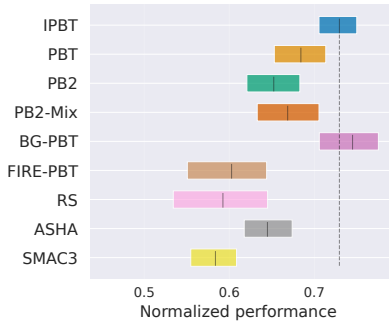


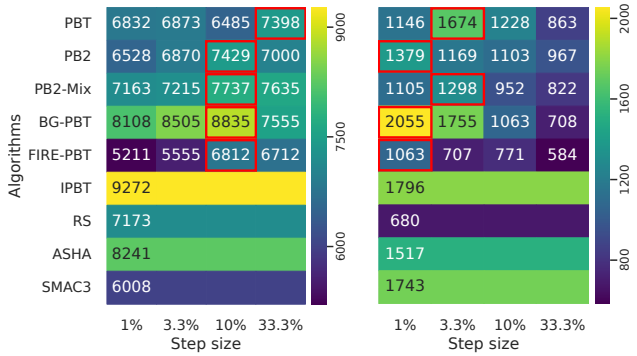
Figure 5: Normalized performance across 8 tasks (IQM and CI) of IPBT, *tuned* PBT variants, and baselines.

step size. The performance of PBT variants relative to each other is reasonable: the most recent Bayesian variant, BG-PBT, achieves the highest IQM, while FIRE-PBT, which uses several subpopulations, performs worst, likely due to the subpopulations being too small.

Selecting the best step size for each task for each PBT variant improves their absolute performance while keeping the relative performance unaffected, as can be seen in Figure 5. Furthermore, selecting the best-performing step size leads to BG-PBT achieving a higher IQM than IPBT, although the difference is not statistically significant. At the same time, IPBT performs statistically significantly better than all other PBT variants, despite using four times less compute (since it was run once, whereas each PBT variant was run four times with different step sizes). We conclude that IPBT should be preferred over other PBT variants, except for BG-PBT, which can achieve better results if its step size is tuned.

The need for tuning step sizes of PBT baselines can be clearly seen in Figure 6, which shows that PBT variants achieve the best performance with larger step sizes (10%, 33.3%) on Humanoid, and smaller (1%, 3.3%) on Hopper.

We further compare IPBT with non-PBT HPO algorithms that can be run out-of-the-box with the same small computation budget. Figure 5 shows that IPBT achieves a much better IQM (statistically signficant) than RS, ASHA, and SMAC3. The unexpectedly poor performance of SMAC3 (on par with RS) led us to investigate, revealing that it is likely due to the algorithm allocating a large fraction of the budget to cheap evaluations and running few expensive ones. This strategy works well for some tasks (e.g, Hopper in Figure 6) but not others (e.g., Humanoid in Figure 6). While HPs of SMAC3 (and ASHA) could potentially be adjusted per-task to achieve



(a) Humanoid (b) Hopper

Figure 6: IQM of scores of PBT variants with different step sizes (the best score of each variant is framed in red), IPBT, and baselines on the Humanoid and Hopper tasks.

better results, this would increase the computational budget and serve as a disadvantage compared to IPBT that performs well without per-task HP adjustment.

4.3 Ablation studies

In this section, we perform ablation studies by varying components of IPBT. The results are shown in Figure 7 and will be discussed in the order presented there.

Firstly, replacing our data-driven mechanism of stagnation detection with the procedure used in BG-PBT reduces IQM, demonstrating the benefit of a more flexible mechanism.

We then investigate components related to the reuse of weight information. Since shrink-perturb is a middle ground between copying weights exactly (which corresponds to parameters $(1.0, 0.0)$) and reinitializing them randomly (which corresponds to $(0.0, 1.0)$), we try both of these options and find that they worsen performance substantially. Next, we vary λ_{shrink} from its default of 0.2 to 0.4 and 0.1, observing a reduction in the IQM that highlights the importance of this HP. Another component related to weight reuse is random reinitialization of weights of half the models on a restart. Disabling it moderately reduces performance, highlighting the importance of injecting randomness to escape local minima. In Appendix F, we additionally replace shrink-perturb with the distillation procedure of BG-PBT on reinforcement learning tasks, and find performance to be similar (which means that performance is not sacrificed when the task-agnostic shrink-perturb is used).

Investigating the HP information reuse, we find that replacing our HP reinitialization strategy based on time-varying BO with either BG-PBT's global BO strategy or random sampling lowers the IQM. However, if no random sampling is done at all, performance decreases, albeit marginally. We conclude that some noise injection improves overall performance.

In IPBT, step size is increased exponentially (doubled) on each restart. Increasing it linearly is worse, while keeping it constant is the worst by a substantial margin. This shows that adjusting the step size is essential to the success of IPBT.

Finally, we investigated whether restarting with twice the

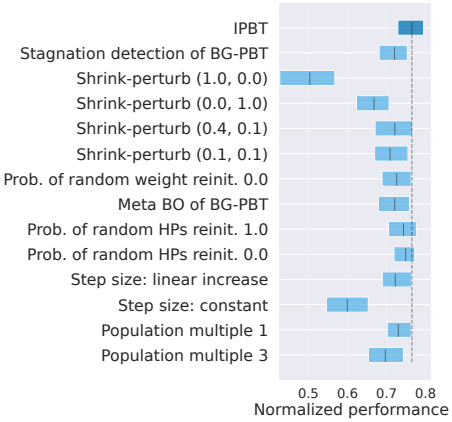


Figure 7: Ablation studies of IPBT: normalized performance (IQM and CI) across 4 tasks.

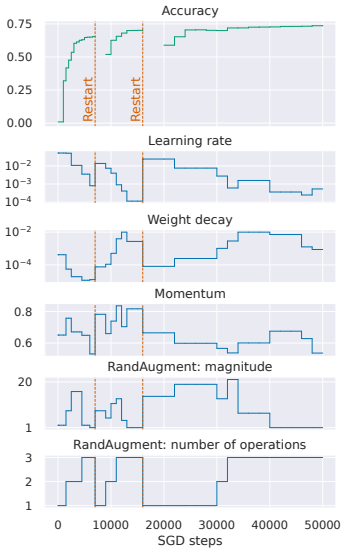


Figure 8: Accuracy and HP schedule of the best solution from an IPBT run on CIFAR-100.

population size (and retaining the best half after the first step) is beneficial compared to not doing so ("Population multiple 1") or starting with thrice the size. Our results indicate that the default value of 2 strikes a good balance between discarding models with poor initial performance and avoiding excessive bias toward models with good performance after a single outer step.

4.4 Qualitatively inspecting a discovered schedule

IPBT is designed to naturally discover schedules with restarts. It can find such schedules not only for the learning rate (where restarts are well-established) but also other hyperparameters, as can be seen in Figure 8. For example, the number of RandAugment[1] operations is reset to 1 on each restart (and then increases within the iteration), while the initial magnitude of the augmentations increases across restarts (and appears to first increase and then decrease during the iteration). This serves as a clear example of how within- and across-iteration optimization both contribute to dynamically optimizing HPs.

5 Discussion and Conclusion

In this paper, we have introduced IPBT, a powerful HPO algorithm that demonstrates superior performance across 8 image classification and reinforcement learning tasks. By automatically adjusting its step size, IPBT substantially outperforms previous PBT variants with untuned step sizes and matches or exceeds their performance when their step sizes are tuned (while avoiding the cost of tuning). Moreover, IPBT significantly surpasses other popular HPO algorithms, such as RS, ASHA, and SMAC3.

IPBT achieves good results thanks to its iterative nature. Restarts allow escape from local optima in both weight and HP space, while efficiency is retained through information reuse across restarts: weights are adapted via shrink-perturb, and the initial HPs of each restart are sampled using a time-varying BO procedure. The decision when (and whether) to trigger a restart is made by our data-driven mechanism, allowing for flexibility based on performance dynamics.

Our ablation studies empirically demonstrated the benefit of the components of IPBT. We have generally observed that injecting some noise both into the weights and into HPs is beneficial in comparison to injecting no noise or relying on random exploration entirely. The component that seems most interesting for further investigation in future work is shrink-perturb (the mechanism for weight reuse), both in terms of how exactly its parameters influence HPO, and whether it can be productively replaced by a different method for partially resetting the weights.

Other promising directions for future work (and current limitations) include adaptation to multi-objective optimization (Dushatskiy et al. 2023) and NAS (Franke et al. 2021).

We designed IPBT to work well for black-box problems and small computation budgets (equivalent to 8 training runs). Algorithms that incorporate prior information about the problem are likely to outperform IPBT if such information is available (Mallik et al. 2023). Similarly, larger computational budgets are likely to enable state-of-the-art BO algorithms (Cowen-Rivers et al. 2022; Yu et al. 2025) to achieve better results.

Nonetheless, enabling efficient search for HP schedules within the wall-clock time of a single training run remains a strong advantage of PBT variants in general and IPBT in particular. Strong out-of-the-box performance of IPBT further marks it as a PBT variant of choice for efficiently optimizing HPs of deep learning tasks in low-budget settings.

[1]RandAugment (Cubuk et al. 2020) is an image augmentation procedure that applies several random augmentations of a specific magnitude (e.g., for rotation, different magnitudes correspond to different maximum angles by which an image can be rotated).

References

Agarwal, R., Schwarzer, M., Castro, P. S., Courville, A. C., and Bellemare, M. G. (2021). Deep Reinforcement Learn-

ing at the Edge of the Statistical Precipice. In Ranzato, M., Beygelzimer, A., Dauphin, Y. N., Liang, P., and Vaughan, J. W., editors, *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*, pages 29304–29320.

Ash, J. T. and Adams, R. P. (2020). On Warm-Starting Neural Network Training. In Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M.-F., and Lin, H.-T., editors, *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*.

Awad, N., Mallik, N., and Hutter, F. (2021). DEHB: Evolutionary Hyberband for Scalable, Robust and Efficient Hyperparameter Optimization. In *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence*, volume 3, pages 2147–2153. ISSN: 1045-0823.

Bergstra, J., Bardenet, R., Bengio, Y., and Kégl, B. (2011). Algorithms for Hyper-Parameter Optimization. In *Advances in Neural Information Processing Systems*, volume 24, pages 2546 – 2554. Curran Associates, Inc.

Bogunovic, I., Scarlett, J., and Cevher, V. (2016). Time-Varying Gaussian Process Bandit Optimization. In *Proceedings of the 19th International Conference on Artificial Intelligence and Statistics*, pages 314–323. PMLR. ISSN: 1938-7228.

Chebykin, A., Alderliesten, T., and Bosman, P. A. N. (2025). To Be Greedy, or Not to Be – That Is the Question for Population Based Training Variants. *Transactions on Machine Learning Research*.

Chebykin, A., Dushatskiy, A., Alderliesten, T., and Bosman, P. A. N. (2023). Shrink-Perturb Improves Architecture Mixing During Population Based Training for Neural Architecture Search. *ECAI 2023 - 26th European Conference on Artificial Intelligence*, 372.

Cowen-Rivers, A. I., Lyu, W., Tutunov, R., Wang, Z., Grosnit, A., Griffiths, R. R., Maraval, A. M., Jianye, H., Wang, J., Peters, J., and Bou-Ammar, H. (2022). HEBO: Pushing The Limits of Sample-Efficient Hyper-parameter Optimisation. *Journal of Artificial Intelligence Research*, 74:1269–1349.

Cubuk, E. D., Zoph, B., Shlens, J., and Le, Q. (2020). RandAugment: Practical Automated Data Augmentation with a Reduced Search Space. In Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M. F., and Lin, H., editors, *Advances in Neural Information Processing Systems*, volume 33, pages 18613–18624. Curran Associates, Inc.

Dalibard, V. and Jaderberg, M. (2021). Faster Improvement Rate Population Based Training. arXiv:2109.13800 [cs, stat].

Davison, A. C. and Hinkley, D. V. (1997). *Bootstrap Methods and Their Application*. Cambridge University press. Number: 1.

Doulazmi, W., Lehuger, A., Toromanoff, M., Charraut, V., Buhet, T., and Moutarde, F. (2025). Multiple-Frequencies Population-Based Training. arXiv:2506.03225 [cs] version: 1.

Dushatskiy, A., Chebykin, A., Alderliesten, T., and Bosman, P. A. N. (2023). Multi-Objective Population Based Training. In *Proceedings of the 40th International Conference on Machine Learning*, pages 8969–8989. PMLR. ISSN: 2640-3498.

Efron, B. and Tibshirani, R. (1993). *An Introduction to the Bootstrap*. Springer.

Elsayed, M., Lan, Q., Lyle, C., and Mahmood, A. R. (2024). Weight Clipping for Deep Continual and Reinforcement Learning. *RLJ*, 5:2198–2217.

Falkner, S., Klein, A., and Hutter, F. (2018). BOHB: Robust and Efficient Hyperparameter Optimization at Scale. In *Proceedings of the 35th International Conference on Machine Learning*, pages 1437–1446. PMLR. ISSN: 2640-3498.

Feurer, M. and Hutter, F. (2019). Hyperparameter optimization. *Automated machine learning: Methods, systems, challenges*, pages 3–33. Publisher: Springer International Publishing.

Franke, J. K. H., Köhler, G., Biedenkapp, A., and Hutter, F. (2021). Sample-Efficient Automated Deep Reinforcement Learning. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net.

Freeman, C. D., Frey, E., Raichuk, A., Girgin, S., Mordatch, I., and Bachem, O. (2021). Brax - A Differentiable Physics Engine for Large Scale Rigid Body Simulation. In *Proceedings of the Thirty-fifth Conference on Neural Information Processing Systems*.

He, K., Zhang, X., Ren, S., and Sun, J. (2016). Deep Residual Learning for Image Recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition 2016*, pages 770–778.

Holm, S. (1979). A Simple Sequentially Rejective Multiple Test Procedure. *Scandinavian Journal of Statistics*, 6(2):65–70.

Jaderberg, M., Dalibard, V., Osindero, S., Czarnecki, W. M., Donahue, J., Razavi, A., Vinyals, O., Green, T., Dunning, I., Simonyan, K., Fernando, C., and Kavukcuoglu, K. (2017). Population Based Training of Neural Networks. *arXiv:1711.09846 [cs]*. arXiv: 1711.09846.

Jamieson, K. and Talwalkar, A. (2016). Non-stochastic Best Arm Identification and Hyperparameter Optimization. In *Proceedings of the 19th International Conference on Artificial Intelligence and Statistics*, pages 240–248. PMLR. ISSN: 1938-7228.

Jankowiak, M., Pleiss, G., and Gardner, J. R. (2020). Parametric Gaussian Process Regressors. In *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*, volume 119 of *Proceedings of Machine Learning Research*, pages 4702–4712. PMLR.

Krizhevsky, A. and Hinton, G. (2009). Learning Multiple Layers of Features from Tiny Images. *Master's thesis, University of Toronto*. Publisher: Toronto, ON, Canada.

Li, L., Jamieson, K., DeSalvo, G., Rostamizadeh, A., and Talwalkar, A. (2018). Hyperband: A Novel Bandit-Based Approach to Hyperparameter Optimization. arXiv:1603.06560 [cs, stat].

Li, L., Jamieson, K. G., Rostamizadeh, A., Gonina, E., Bentzur, J., Hardt, M., Recht, B., and Talwalkar, A. (2020). A System for Massively Parallel Hyperparameter Tuning. In Dhillon, I. S., Papailiopoulos, D. S., and Sze, V., editors, *Proceedings of the Third Conference on Machine Learning and Systems, MLSys 2020, Austin, TX, USA, March 2-4, 2020*. mlsys.org.

Lindauer, M., Eggensperger, K., Feurer, M., Biedenkapp, A., Deng, D., Benjamins, C., Ruhkopf, T., Sass, R., and Hutter, F. (2022). SMAC3: A Versatile Bayesian Optimization Package for Hyperparameter Optimization. *Journal of Machine Learning Research*, 23(54):1–9.

Liu, Z., Mao, H., Wu, C.-Y., Feichtenhofer, C., Darrell, T., and Xie, S. (2022). A convnet for the 2020s. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 11976–11986.

Loshchilov, I. and Hutter, F. (2017). SGDR: Stochastic Gradient Descent with Warm Restarts. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net.

Lourenço, H. R., Martin, O. C., and Stützle, T. (2003). Iterated Local Search. In *Handbook of Metaheuristics*, pages 320–353. Springer.

Mallik, N., Bergman, E., Hvarfner, C., Stoll, D., Janowski, M., Lindauer, M., Nardi, L., and Hutter, F. (2023). PriorBand: Practical Hyperparameter Optimization in the Age of Deep Learning. In *Thirty-seventh Conference on Neural Information Processing Systems, 2023*.

Parker-Holder, J., Nguyen, V., Desai, S., and Roberts, S. J. (2021). Tuning Mixed Input Hyperparameters on the Fly for Efficient Population Based AutoRL. In *Advances in Neural Information Processing Systems*, volume 34, pages 15513–15528. Curran Associates, Inc.

Parker-Holder, J., Nguyen, V., and Roberts, S. J. (2020). Provably Efficient Online Hyperparameter Optimization with Population-Based Bandits. In Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M. F., and Lin, H., editors, *Advances in Neural Information Processing Systems*, volume 33, pages 17200–17211. Curran Associates, Inc.

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., and Klimov, O. (2017). Proximal Policy Optimization Algorithms. arXiv:1707.06347 [cs].

Snoek, J., Larochelle, H., and Adams, R. P. (2012). Practical Bayesian Optimization of Machine Learning Algorithms. In *Advances in Neural Information Processing Systems*, volume 25. Curran Associates, Inc.

Sokar, G., Agarwal, R., Castro, P. S., and Evci, U. (2023). The Dormant Neuron Phenomenon in Deep Reinforcement Learning. In *Proceedings of the 40th International Conference on Machine Learning*, pages 32145–32168. PMLR. ISSN: 2640-3498.

Stanford CS231N (2017). Tiny ImageNet.

Tan, J. M., Liao, H., Liu, W., Fan, C., Huang, J., Liu, Z., Yan, J., Tan, J. M., Liao, H., Liu, W., Fan, C., Huang, J., Liu, Z., and Yan, J. (2024). Hyperparameter Optimization: Classics, Acceleration, Online, Multi-Objective, and Tools. *Mathematical Biosciences and Engineering*, 21(6):6289–6335.

Tan, M. and Le, Q. (2021). EfficientNetV2: Smaller Models and Faster Training. In *Proceedings of the 38th International Conference on Machine Learning*, pages 10096–10106. PMLR. ISSN: 2640-3498.

Wan, X., Lu, C., Parker-Holder, J., Ball, P. J., Nguyen, V., Ru, B., and Osborne, M. (2022). Bayesian Generational Population-Based Training. In *Proceedings of the First International Conference on Automated Machine Learning*, pages 14/1–27. PMLR. ISSN: 2640-3498.

Wan, X., Nguyen, V., Ha, H., Ru, B., Lu, C., and Osborne, M. A. (2021). Think Global and Act Local: Bayesian Optimisation over High-Dimensional Categorical and Mixed Search Spaces. In *Proceedings of the 38th International Conference on Machine Learning*, pages 10663–10674. PMLR. ISSN: 2640-3498.

Won, J., Lee, H.-S., and Lee, J.-W. (2025). A Review on Multi-fidelity Hyperparameter Optimization in Machine Learning. *ICT Express*, 11(2):245–257.

Xiao, H., Rasul, K., and Vollgraf, R. (2017). Fashion-MNIST: a Novel Image Dataset for Benchmarking Machine Learning Algorithms. arXiv:1708.07747 [cs, stat].

Yu, R. T.-Y., Picard, C., and Ahmed, F. (2025). GIT-BO: High-Dimensional Bayesian Optimization with Tabular Foundation Models. In *1st ICML Workshop on Foundation Models for Structured Data, 2025*.

Zaidi, S., Berariu, T., Kim, H., Bornschein, J., Clopath, C., Teh, Y. W., and Pascanu, R. (2023). When Does Reinitialization Work? In *Proceedings on "I Can't Believe It's Not Better! - Understanding Deep Learning Through Empirical Falsification" at NeurIPS 2022 Workshops*, pages 12–26. PMLR. ISSN: 2640-3498.

Appendix

A Pseudocode of IPBT

Algorithm 1: IPBT

Input: population size N, maximum number of inner steps $\text{step}_{\max}$, step size $\text{step}_{\text{inner}}$, selection parameter λ

pop $\leftarrow$ {a pair of random HPs $\boldsymbol{h}^i$ and weights θ^i}$_{i=0}^{N-1}$
D $\leftarrow$ {} // Dataset of performance differences
$\text{D}_{\text{it}} \leftarrow$ {} // Dataset of performance differences within the current iteration (i.e., before a restart)
step $\leftarrow 0$
while step $<$ $\text{step}_{\max}$ **do**
  step $\leftarrow$ step $+$ $\text{step}_{\text{inner}}$
  for $i \leftarrow 0$ **to** $N-1$ **do** // in parallel
    Train θ^i for $\text{step}_{\text{inner}}$ steps using $\boldsymbol{h}^i$
    $\text{perf}_{\text{i}}^{\text{step}} \leftarrow \texttt{evaluate}(\theta^i)$
    $\text{info}_{\text{i}} \leftarrow \left(\text{perf}_{\text{i}}^{\text{step}} - \text{perf}_{\text{i}}^{\text{step}-\text{step}_{\text{inner}}}, \text{step}, \boldsymbol{h}^i\right)$ // store difference to the previous performance and the HPs
    $\text{D}_{\text{it}} \leftarrow \text{D}_{\text{it}} \cup \text{info}_{\text{i}}$
    D $\leftarrow$ D $\cup$ info_{i}
  end for
  Check if a restart is triggered // Section 3.2
  if no restart **then**
    Sort pop by perf
    Replace the worst $\lambda\%$ with the copies of the best $\lambda\%$ (selected randomly)
    Fit a surrogate on D_{it}, use BO to set HPs of the copies
  else
    Reinitialize weights // Section 3.3
    Reinitialize HPs using BO on D // Section 3.3
    $\text{step}_{\text{inner}} \leftarrow 2 \cdot \text{step}_{\text{inner}}$ // Section 3.4
    $\text{D}_{\text{it}} \leftarrow$ {}
  end if
end while

The pseudocode of IPBT is listed in Algorithm 1, see the main text and the source code for further details.

B Tasks, search spaces, and evaluation

Our experiments are run on the tasks previously used for evaluating PBT variants (Chebykin et al. 2025). In image classification tasks, hyperparameters of a ResNet-12 (He et al. 2016) are optimized on CIFAR-10/100 (Krizhevsky and Hinton 2009), and Fashion-MNIST (Xiao et al. 2017) datasets for 50,000 Stochastic Gradient Descent (SGD) steps; and hyperparameters of a ConvNeXt-T (Liu et al. 2022) are optimized on Tiny ImageNet (Stanford CS231N 2017) for 150,000 steps. In reinforcement learning tasks, hyperparameters of a proximal policy optimization (Schulman et al. 2017) agent are optimized on tasks implemented in Brax (Freeman et al. 2021) (Humanoid, Hopper, Pusher, Walker) during 150,000,000 environment interactions.

For PBT variants, the *large* search spaces of (Chebykin et al. 2025) are used for both classification (Table 1) and reinforcement learning (Table 2) tasks. For the non-PBT baselines, the search spaces are extended with hyperparameters of a cosine learning rate schedule with restarts (Table 3).

We did not change evaluation metrics, using accuracy for image classification, and task score for reinforcement learning tasks.

All performance claims in the paper are based on the test performances of the best models of each run of an algorithm. For algorithms with restarts (IPBT, BG-PBT, non-PBT baselines), the best model is chosen based on validation performance achieved either before a restart or at the end of training, while for the PBT variants without restarts, the best model is chosen based on the validation performance at the end of training.

C Statistical testing

In the main text, in order to compare algorithms while taking performance of all tasks into account, the IQM of the normalized performance across tasks and seeds is estimated via stratified bootstrapping (Agarwal et al. 2021). To assess statistical significance of differences between the IQMs of two algorithms, a paired stratified bootstrap test (Efron and Tibshirani 1993) is executed with 50,000 replicates.

In every replicate, we sample seeds with replacement, use those seeds as indices for both algorithms across all tasks (thus getting a paired sample), and compute the difference between the IQMs. The replicate differences are then centered by the observed IQM, and a two-sided p-value is obtained from the (corrected) proportion of centered replicates whose magnitude exceeds the observed difference (Davison and Hinkley 1997). The code for this procedure is included in our source code release.

IPBT is pairwise compared with 5 tuned PBT variants and 3 baselines (RS, ASHA, SMAC3) — 8 tests in total. The target significance level is 0.05, and the Holm correction (Holm 1979) is used to control the family-wise error rate. The results in Table 4 show that performance of IPBT is significantly different from that of all algorithms except for BG-PBT.

D Hyperparameters

The hyperparameters of all the algorithms are listed in the configuration files included in the source code release. In this section, we discuss the setting of hyperparameters of novel components of IPBT.

Shrink-perturb The parameter values used in the main experiments are $(0.2, 0.1)$. The results when using $(0.4, 0.1)$ and $(0.1, 0.1)$ are reported in the ablation study in the main text. Additionally, the setting $(0.5, 0.5)$ was tried during early development (on the CIFAR-10 task) and found to perform worse.

Stagnation detection The parameters for restart detection used in the experiments were set to $t_{\text{patience}} = 3$ and $t_{\text{interval}} = 15$. These values were chosen based on visual inspection of performance traces during preliminary experiments. While t_{interval} was not varied, values of 2, 3, and 4 were tested for t_{patience} using an early version of the al-

Hyperparameter	Type	Exponent base	Range
Learning rate	real	10	[-6, 0]
Number of augmentations	integer	✗	[1, 4]
Augmentation strength	integer	✗	[1, 30]
Weight decay	real	10	[-8, -2]
Momentum	real	✗	[0.5, 0.999]

Table 1: Search space for the classification tasks (the *large* search space from (Chebykin et al. 2025)).

Hyperparameter	Type	Exponent base	Range
Learning rate	real	10	[-4, -3]
Entropy coefficient	real	10	[-6, -1]
Batch size	integer	2	[8, 10]
Discount factor	real	✗	[0.9, 0.9999]
Unroll length	integer	✗	[5, 15]
Reward scaling	real	✗	[0.05, 20]
Number of updates per epoch	integer	✗	[2, 16]
GAE parameter	real	✗	[0.9, 1]
Clipping parameter	real	✗	[0.1, 0.4]

Table 2: Search space for the reinforcement learning tasks (the *large* search space from (Chebykin et al. 2025)).

Hyperparameter	Type	Exponent base	Range
Number of iterations before the first restart	integer	✗	[5, 50]
Multiplier for the number of iterations between restarts	integer	✗	[1, 2]
Minimum learning rate	real	10	[-8, -6]

Table 3: Additional hyperparameters of a cosine learning rate schedule with restarts used by non-PBT baselines. Note that the non-PBT baselines are run with an iteration equal to 1% of the budget, thus the number of iterations before the first restart is between 5% and 50% of the budget.

Table 4: P-values (rounded to 5 decimal places) of pair-wise statistical tests of IPBT against baselines (sorted by p-value)

Algorithm	Rejected H_0	p	p (Holm)
FIRE-PBT	Yes	0.00002	0.00016
SMAC3	Yes	0.00002	0.00016
Random search	Yes	0.00004	0.00024
ASHA	Yes	0.00012	0.00060
PB2	Yes	0.00022	0.00088
PB2-Mix	Yes	0.00810	0.02430
PBT	Yes	0.02070	0.04140
BG-PBT	No	0.49701	0.49701

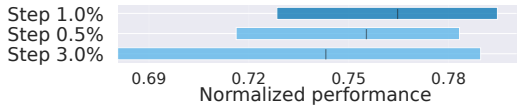


Figure 9: Extra ablation study on the initial step size of IPBT: normalized performance (IQM and CI) across 4 tasks (same setup as the ablation study in the main text).

gorithm on CIFAR-10/100 and Brax Humanoid tasks. The setting $t_{\text{patience}} = 3$ yielded the highest IQM.

Initial step size In the experiments, the initial step size is set to 1% of the budget. We additionally experiment with settings of 0.5% and 3%, results are shown in an ablation study in Figure 9 (not included in the main text due to space constraints). It can be observed that reducing the step size to 0.5% leads to only a slight decrease in IQM, as IPBT can increase the step size when needed. Starting with a larger step size of 3% reduces performance more, as IPBT is not designed to decrease the step size and larger steps lead to faster budget exhaustion.

Strategy of step size increase The step size is doubled on each restart. We additionally tried increasing it linearly, with the results reported in the ablation study in the main text.

All preliminary experiments were run with seeds distinct from those used for the ablation studies and the main experiments.

E Implementation details

Hardware and software The configuration of the servers used to run experiments: 3 Nvidia A5000 GPUs, 2 Intel(R) Xeon(R) Bronze 3206R CPUs, and 96 GB of RAM. The OS is Fedora Linux 36. The random seeds and the versions of all used frameworks and libraries are specified in the configuration files included in the source code release.

SMAC3 Directly using the latest version of SMAC3 (2.3.1) and setting the total budget in line with the documentation caused the algorithm to stop after only $\approx 75\%$ of the budget was used. We believe this issue stems from a discrepancy between the theoretically calculated budget usage, and the practical asynchronous implementation. To address this, we forked the code and introduced an alternative stopping criterion that monitors the actual budget consumed. This resulted

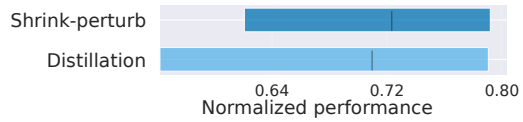


Figure 10: Extra ablation study on the weight reuse mechanism of IPBT: normalized performance (IQM and CI) across the Humanoid and Hopper tasks.

in nearly full budget usage and improved performance on CIFAR-10 in preliminary experiments.

BG-PBT As mentioned in Section 3.3, Wan et al. (2022) use either 26.6% or 40% of the budget in their second stopping criterion (depending on the task). For uniformity, we use 30% for all tasks. We enable this criterion only with the step size of 1% (that is the closest to the original setting of BG-PBT) because otherwise too few steps are done before a forced restart is triggered. We also enable distillation only in this setting. Additionally, when the step size is 33.3%, we change $N_{\text{multiplier}}$ from the default 3 to 1 because otherwise the entire budget is spent on random initialization.

F Replacing shrink-perturb with distillation

In BG-PBT, a distillation procedure is used to transfer information stored in weights across restarts. We replace shrink-perturb in IPBT with this procedure, and evaluate it on the Humanoid and Hopper tasks (as the distillation is specific to reinforcement learning tasks). Figure 10 shows that the performance is approximately the same, meaning that task-agnostic shrink-perturb matches task-specific distillation.